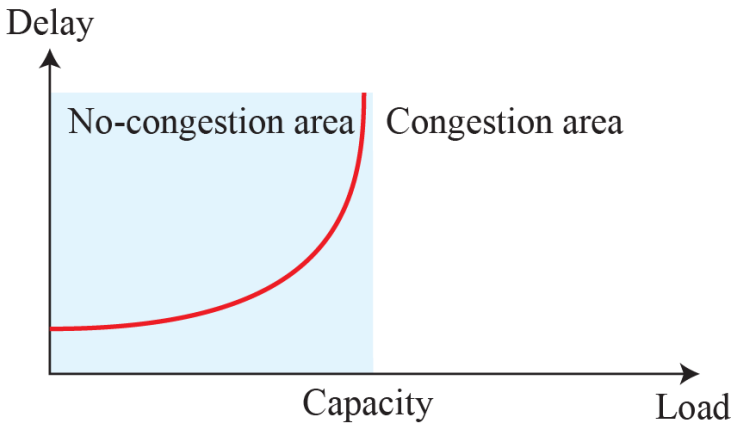
Quality of service in packet networks

Best effort Internet

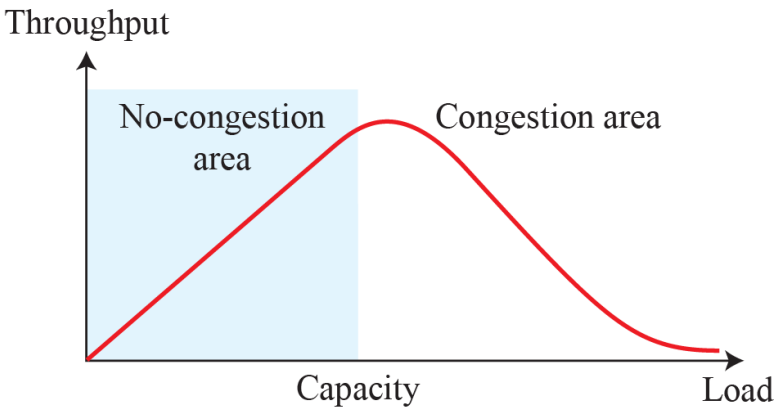
- The Internet is an IP switched network that makes use of optical lightpaths for point to point transmission (i.e., communications between routers).
- While lightpaths are highly reliable (they start and terminate a link), routers are based on bets effort (packets can get lost due to congestion)
- The Internet was born as a best effort service:
  - It does its best at transferring data between points, but it does not provide guarantees in terms of: packet loss, latency (or delay), jitter and capacity
  - Packet loss: the fact that a packet sent by an application does not get to its destination [can be measured as a rate of pkt lost/pkt sent]
  - Latency: the time it takes for a packet to reach its destination [measured in ms]
  - Jitter: the variation in the latency of received packets [can be measured as root mean square]
  - Capacity: how many bits (that are packed into packets) can be received per second [measured in b/s (or multiples)]

Is internet unreliable?

- Yes, although layer 4 (transport layer), with TCP, creates virtual reliability on top of an unreliable network
  - By retransmitting lost packets and reordering them reduces the issue due to packet loss
  - In addition, it operates a congestion control system that reduces the transmission rate when congestions is sensed, to try and reduce congestion.
- OK, but does this solve the packet loss issue?
  - If you have ever experienced an application or webpage being too slow to be any use, then you have seen that TCP cannot solve everything
- And what about latency, jitter, capacity? Are they a real problem?



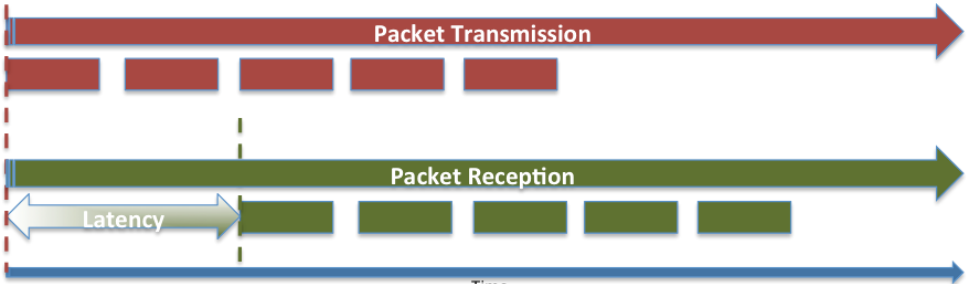
a. Delay as a function of load



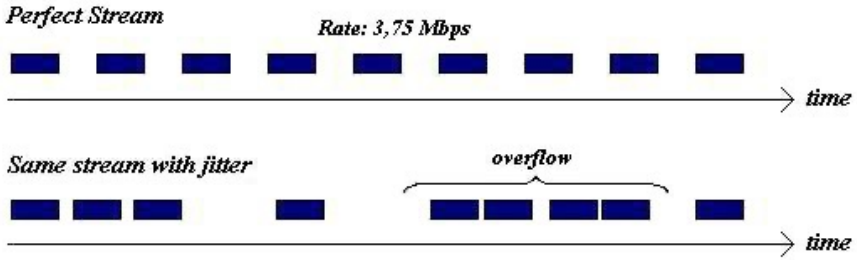
b. Throughput as a function of load

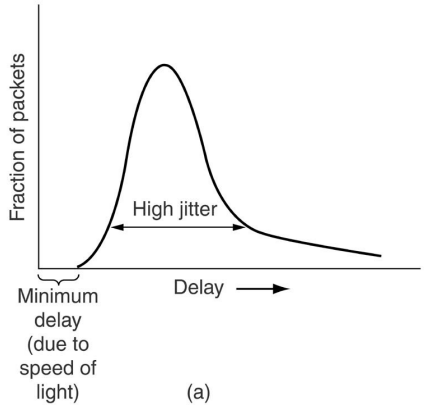
Latency and Jitter

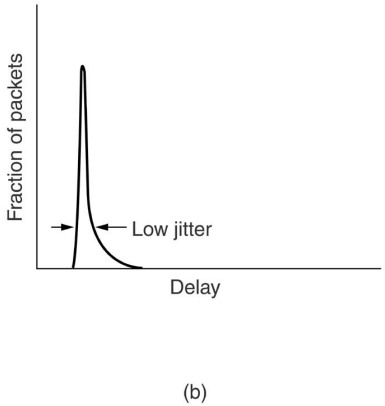
- Latency:
  - is a problem typically for voice or video conferencing, e.g., when there are two parties communicating that need to receive a message within a short time.
  - cannot be solved above layer 4, but needs to be tackled at layer 3 and below (shorter routes, lower queuing delay in routers/switches)



- Jitter:
  - is a problem for some real time applications that expect a packet within a certain time from the previous one, like VoiP and video.
  - can be solved at application level through buffers, but these increase latency







Packet loss and capacity

- Packet loss:
  - In principle it affects all applications, but TCP is capable of at least recovering the lost packets:
    - Unrecovered packet loss is catastrophic for file (document, application,…) transfer (if a piece is missing the file is corrupted, some info is lost and the file might not even open)
    - Not too bad for voice or video (especially streaming), where you only loose one piece. Indeed voice and video conferencing don't use TCP as it's better to loose a frame than having it back at a later time.

- Capacity:
  - Capacity relates to all the metrics above. As not enough capacity will cause packet loss, increased latency and jitter.
  - real time applications need a certain capacity otherwise they cannot work (VoIP, video, interactive applications)… anything that cannot be downloaded completely in advance
    - Video could in principle be downloaded in advance, but if the capacity is low you'd be waiting a long time…
  - capacity is a layer 1 issue, although in a network it's limited by the slowest link, so also layers 2 and 3 play an important role

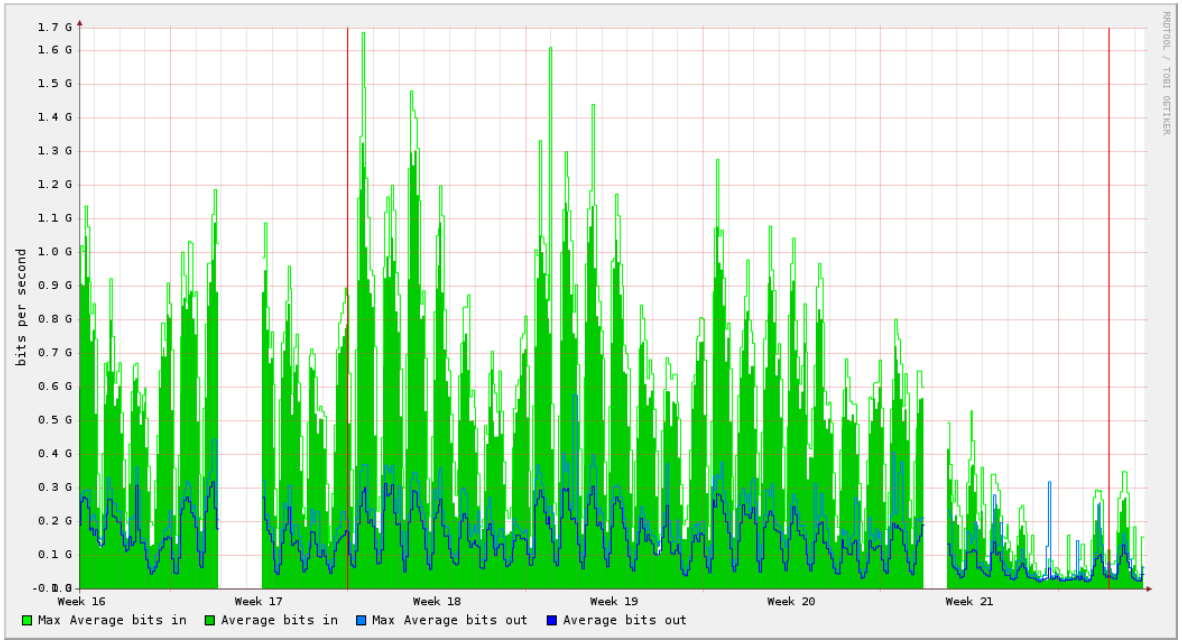
Is best effort good enough?

- Not every application is the same:
  - Some don't work well as best effort
- Not every person is the same:
  - The same application behaviour can be perceived differently by different people

- As the Internet has become a commodity and as more and more of our life depends on it (every aspect, from work, college, entertainment, …) we rely on it
  - the idea of things working as best effort is less and less accepted
  - plus, some applications can be quite critical (emergency services), and a large part of these have to implement other means of communications that are more reliable

Solutions

- One way of thinking is that ultimately all depends on capacity…
- … if capacity is enough, there is no or little queuing at routers/switches, so no congestion
  - small delay, small jitter and no packet loss



- But in an environment like the Internet with large variation in link occupancy, it is prohibitively expensive to overprovision the network so that all congestions are avoided at all times

Statistical multiplexing

- The Internet is based on statistical multiplexing of packets, which is an efficient method of sharing resources, when data is bursty

- Is there a way to maintain the benefit of statistical multiplexing (thus without excessive overprovisioning), and still have good behaviour in terms of packet loss, latency and jitter?

- The key is in the fact that not all applications have the same requirement:
  - A VoIP call or video conference needs strict assurance of packet loss, latency and jitter, while a file transfer is much less sensitive to latency and jitter and in part to packet loss.
  - So the key idea is that of prioritisation

QoS = prioritization!!!

- QoS is often (and correctly) referred to as QoS differentiation
  - If we do statistical multiplexing at times things will get congested
  - When that happens QoS gives priority to packets form applications that would suffer most from increase in packet loss, latency and jitter

How does it work

- Packet headers typically have a field that can indicate the priority:
  - Ethernet *Priority code point (PCP)* in the VLAN tag
  - IP *Differentiated Services Code Point (DSCP)*
  - MPLS *Traffic Class (TC)*

- Based on this fields, when congestion occurs routers/switches can give priority to packets with higher priority values (i.e. higher number)

- Do you see any flaw in this??
- …aka the tragedy of the commons…
- …why would I not write an app that always indicates the highest priority for all of my traffic, at the end I have a flat rate contract…

Who sets the priority bits?

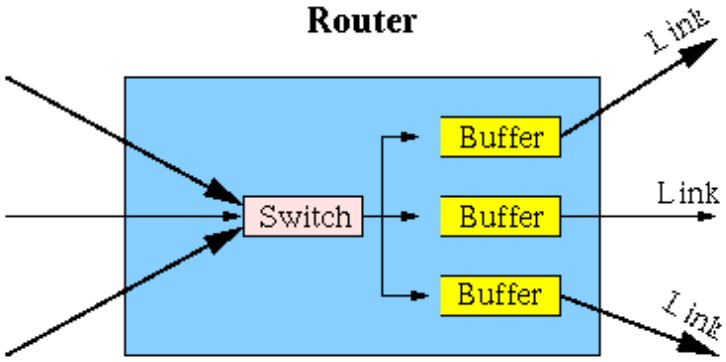
- If the application does that, can you trust the application?
  - I could re-write any application so that it marks packets as highest priority…
  - … or even write a program that changes the values of every packet to highest priority
- What happens if everyone does that?
  - Since priority is a relative concept (i.e. prioritise with respect to other packets), if everyone has the same highest priority we go back to best effort for all packets…!!

So, is QoS used at all?

- Yes it is but typically user applications are untrusted
- The priority bits might be decided by the entity purchasing the service (here we consider business users) but are then checked by the Network Provider offering the service
- Notice that for example not all VoIP is the same, e.g., VoIP from an operator phone network is given priority, unlike skype or others when on a public network…
- Let’s see how it works in practice

Let's take a router

- Let's take into consideration a router
  - Packets come in
  - A routing table check the destination address and puts them in a queue at the correct output port
  - Packets at the head of a queue gets out first (FIFO queues)



QoS tool

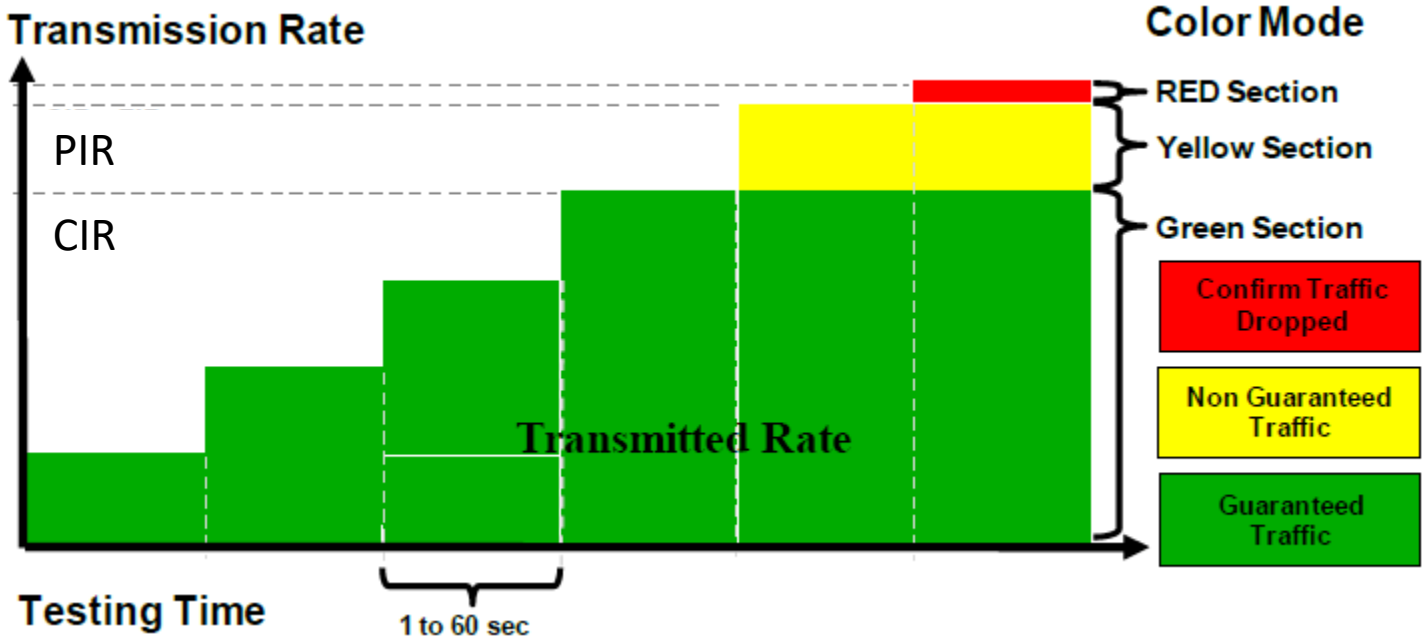
- QoS is based on a number of tools that can be combined together:
  - Policer: discards packets when they go above a pre-established threshold
  - Shaper: delay packets so that the bandwidth threshold is not exceeded at any given time
  - Classifier: inspects an incoming packet and assigns to it a class of service (COS)
  - Metering and coloring: check the rate at which packets are coming in against pre-defined thresholds and subsequently marks packets with different “colors”
  - Queuing differentiation: queues all work in FIFO mode, but packets with different COS can be assigned different queues
  - Scheduler: it decides in which order to get packets from the different queues
  - Rewrite: it can modify a packet COS marking

Classifier

- The classifiers inspects a packet (e.g., could look at incoming port, address, COS marking,…) and assigns it a COS class inside the router
- In an ideal environment with full trust between all parties, the router should just look at the COS marking in the packet and assign the corresponding COS class accordingly.
- In practice, if a packet has COS marking three actions are possible:
  - Trust it and use it for its internal router operations
  - Increase its granularity (for example provide more in depth differentiation)
  - Change it completely

Metering and coloring

- A customer could have a contract with an operator through a Service Level Agreement (SLA)
  - The contract will say that the link from customer to operator has a certain capacity of:
    - 5 Gb/s of Committed Information Rate (must be satisfied)
    - 8 Gb/s of Peak Information Rate (is satisfied if capacity is available)
    - Additional details on how it is measured, any level of availability and protection and eventual penalties

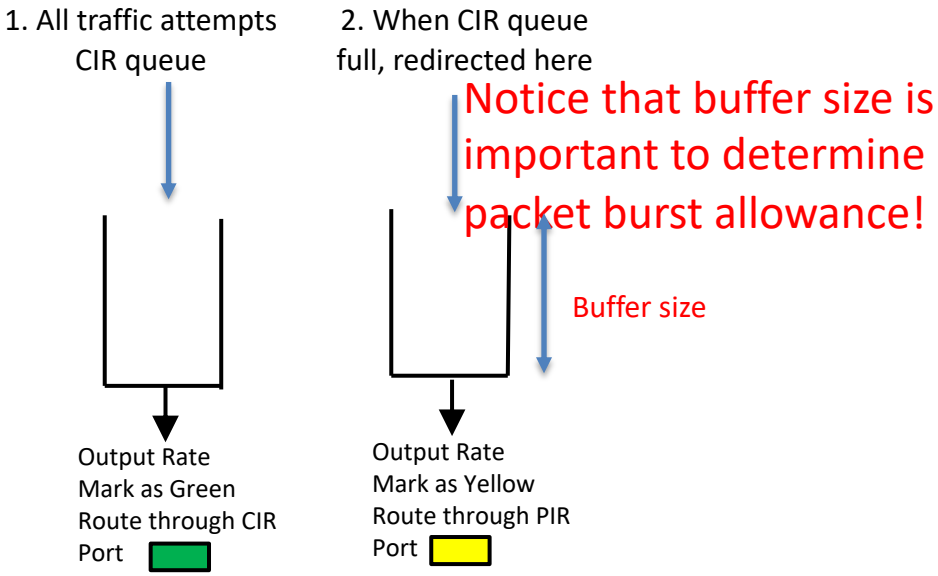


Two rate Three Colour Marking (TrTCM)

Color blind

All input traffic is the same:

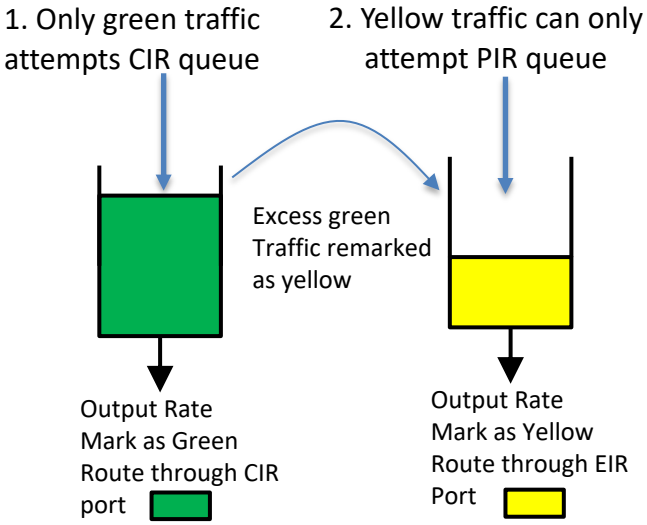
- Data below CIR marked green
- Above CIR, but below PIR as yellow
- Above PIR as red (dropped)



Color aware

Input already marked (green, yellow)

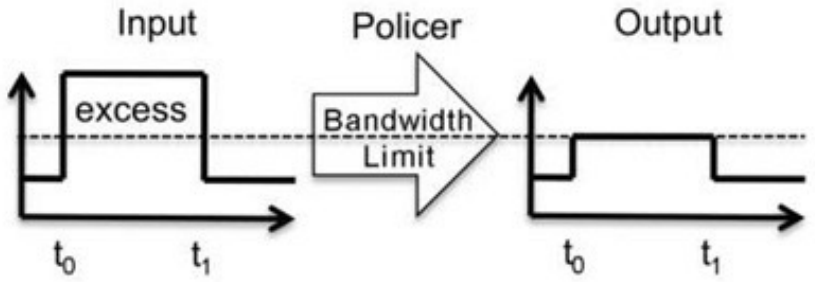
- Green goes into CIR queue, but if above CIR is sent to PIR queue
- Yellow goes into PIR queue
- Any traffic above PIR is red (dropped)
- Notice that yellow never becomes green even if CIR still available



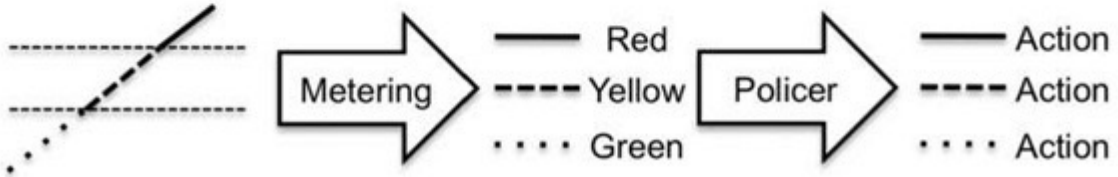
Queue mechanisms are implemented through token bucket (see slide 21)

Policer

- A policer performs an action on packets based on a data rate threshold limit
- Any traffic above a certain rate is automatically dropped, typically performed at the ingress of a router

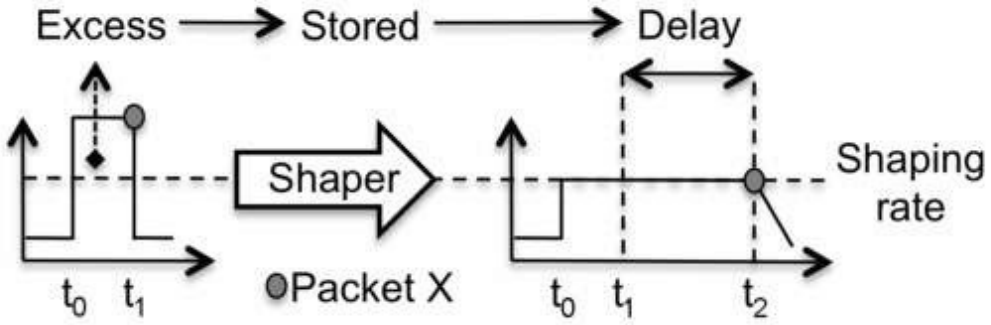


- It can be considered as an action following the metering for red-marked packets, at the input port.

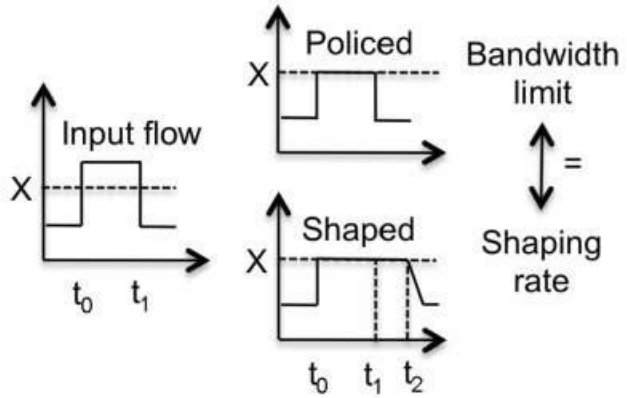


Shaper

- A shaper also performs an action on packets based on a data rate threshold limit
- However the action is not to drop but to delay, so that the threshold is not exceeded
- This can be used for example at an output port

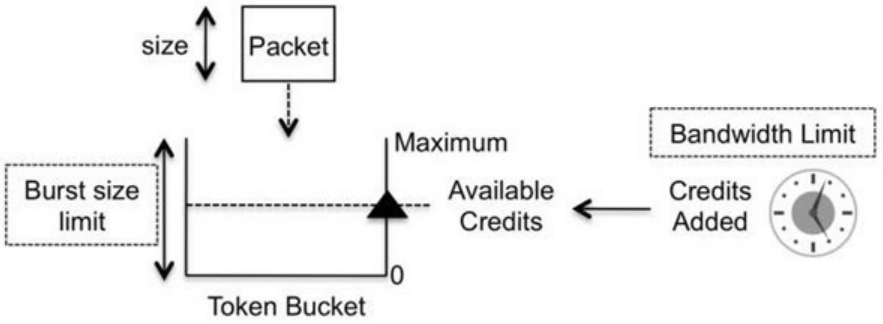


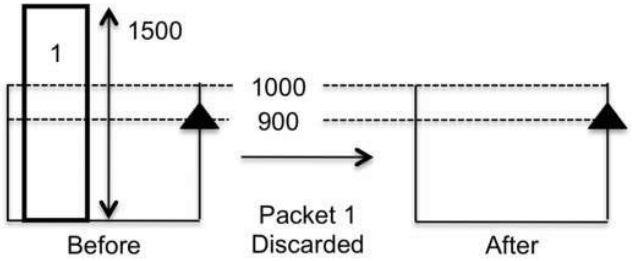
- Shaper vs policer:



Policer implementation: token bucket

- The idea is that tokens (credits) are added to the system periodically and each packet coming in has two options:
  - If there are credits available it enters one of the queues
  - If there aren't credits it gest discarded (and credits are not consumed)

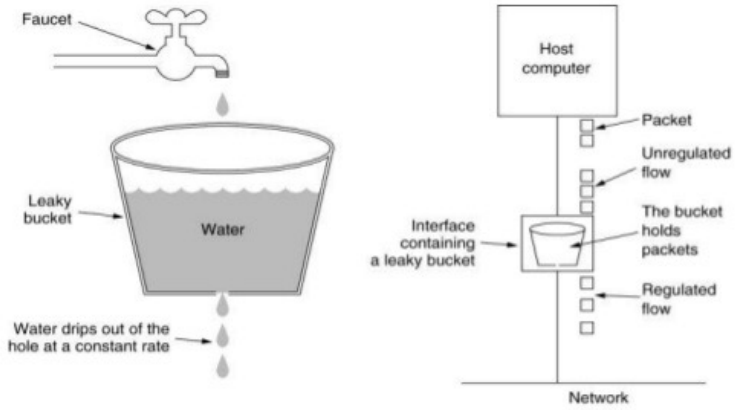




- The depth of the bucket determines the maximum number of credits that can be stored at any one time
  - This determines the burst size limit
- The number of tokens per time determines instead the bandwidth limit value

Shaper implementation: leaky bucket

- The idea is to have a buffer where incoming packets are stored that accepts new packets at whatever rate they come in
- The buffer is then depleted at a constant rate



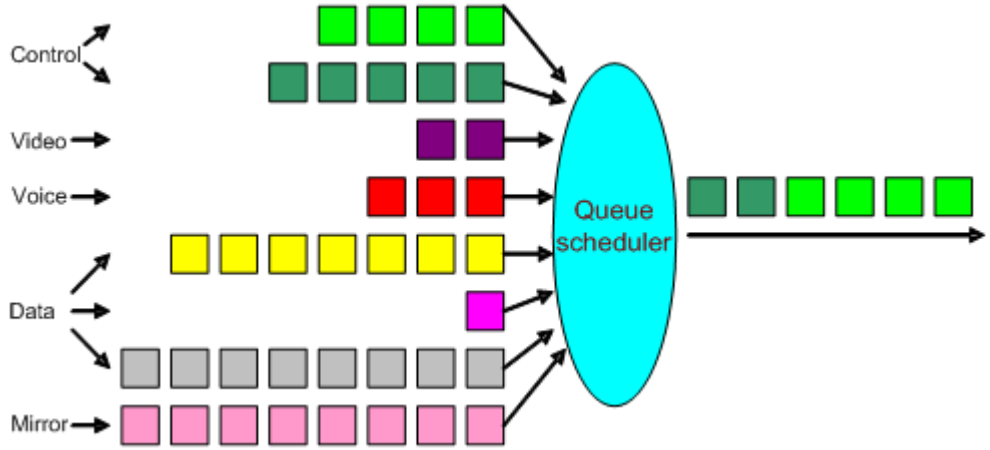
- The rate at which packets taken from the buffer determines the bandwidth limit
- The size of the buffer (bucket) determines for how long excess traffic can be sent before the buffer overflows and starts discarding incoming packets
- Notice that the ability to keep the traffic rather than discarding it (as in the policer) is at the expense of increased latency.

Queuing

- Queues work in FIFO mode and packets proceed linearly from input to output:
  - Packets cannot pass other packets once they're in the queue
- However packets can be dropped before entering a queue based on the queue fill level.
  - If a queue is 100% full then all new packets towards that queue are dropped (there is not physical place where to store them)
  - However there are other actions possible, to discard some packets before a queue is 100% full
- Depending on classification, packets can be sent to different queues
- Typically a router (or switch) port has up to 8 different queues per port (Ethernet PCP, MPLS TC have 3 bits – i.e. can identify up to 8 different classes, DSCP up to 64)

Scheduler

- The scheduler it's associated with the use of differentiated queues, as it implements an algorithm to decide how to serve the queues
- Many options are possible:
  - A simple way is to serve first all packets in the queue with highest priority…
  - … then the queue with second higher priority… and so on
  - However also more complex behaviours are possible



Rewrite

- It changes the COS marking of an incoming packet
- One straightforward reason is if the router does not trust the entity that has marked the packet:
  - It carries out its own classification and assigns a new marking
- However can also be used to signal information to a downstream router:
  - E.g., two packets, A and B, arrive with a trusted marking of ‘1’
  - However A is marked as green and B as yellow
  - The router can change the marking of B from ‘1’ to ‘2’ to signal the downstream router that packet B had exceeded the CIR rate
  - This implies that both router have a similar interpretation of the COS marking

Effects of QoS tools on packet loss, latency and jitter

- Depending on the traffic classification the QoS tools are those that introduce packet loss, delay and jitter
- Packet loss caused by:
  - Traffic policer, drop by the queue
- Latency cause by:
  - Shaper, scheduler
- Jitter cause by:
  - Shaper, scheduler

Effect of queue size on delay and jitter

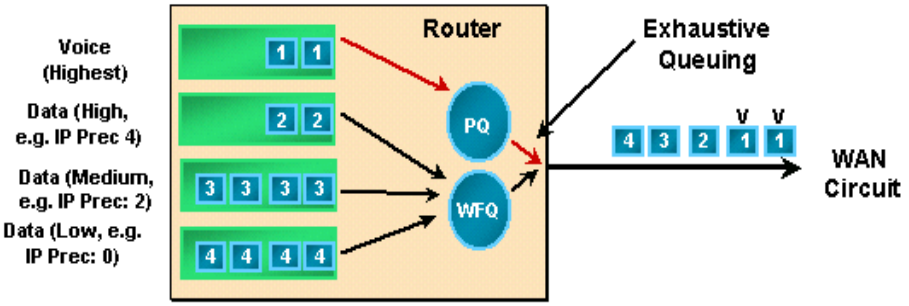
- Queue size affect packet loss, delay and jitter
- A longer queue is able to absorb heavier traffic bursts:
  - Lower chance of queue filling up and traffic being dropped
  - However packets spend longer in the queue and thus the delay increases (and so the jitter)
    - For some applications it makes sense to drop a packet rather than having too high a delay

Schedulers implementation

- Given one or more queues for one output port, where packets are inserted depending on their COS value, a scheduler implements an algorithm to decide how to take packets from the different queues for transmission

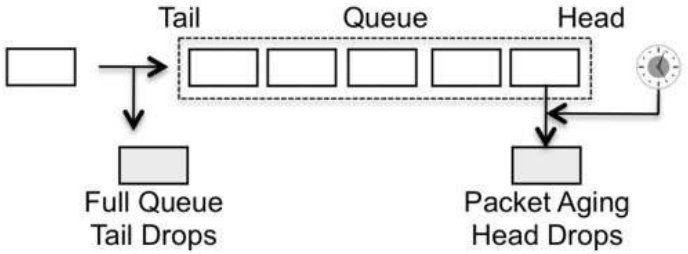
- Many algorithms have been developed, some of which are:

- FIFO queuing
- Fair queuing (FQ)
- Strict Priority queuing (SP)
- Weighted fair queuing (WFQ)
- Weighted round robin (WRR)
- Deficit weight round robin (DWRR)
- Priority-based deficit weighted round robin (PB-DWRR)

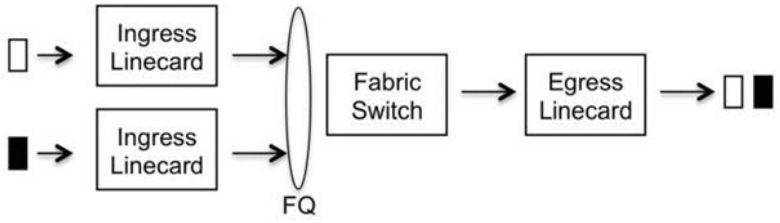


FIFO and FQ

- FIFO is a simple queue with no service differentiation.
  - It preserves packet order
  - When full it drops packets

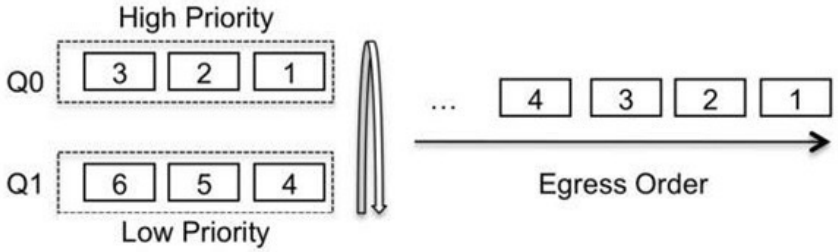


- Fair Queuing:
  - It isolates different flows into different logical queues
  - The scheduler picks a packets from the logical queue in round robin.
  - It does not provide QoS differentiation because it's fair to all flows
  - Not used in practice because separating flows is too computationally intensive

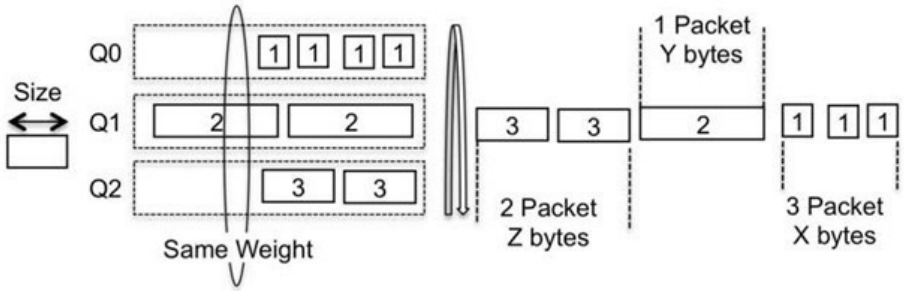


Strict priority (SP) and WFQ

- Strict priority queuing has multiple queues of different priority and packets are taken first from the highest priority queue, then from second highest, and so on:
  - The priority is strict as Q1 is not served until Q0 is empty
  - Low priority traffic can become stalled (e.g., TCP sessions expiring) id higher priority queues fill in all the capacity

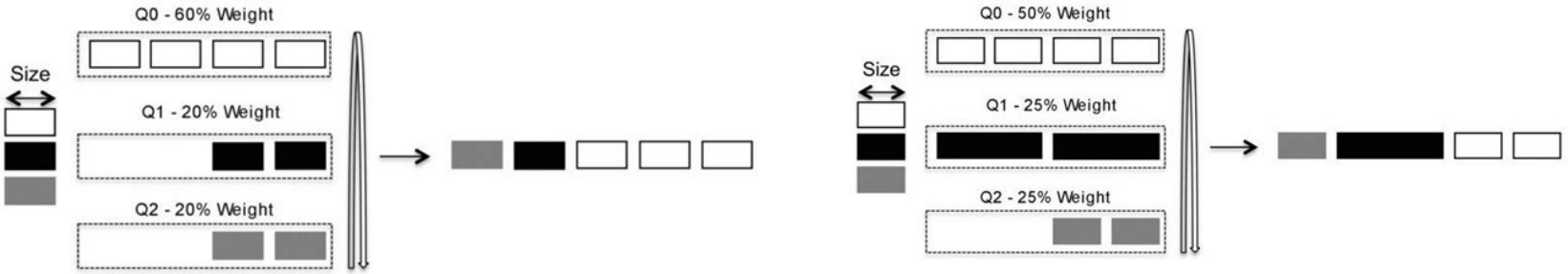


- Weighted fair queueing assigns proportional weights rather than giving strict priority to queues:
  - For example Q1 is served less often than Q0, but it doesn't have to wait until Q0 is empty
  - It also operates bit-by-bit to avoid that flows with larger packets get to transmit more data.
    - E.g., in a case where queues have same weight, a queue with smaller packets will transmit more



WRR and DWRR

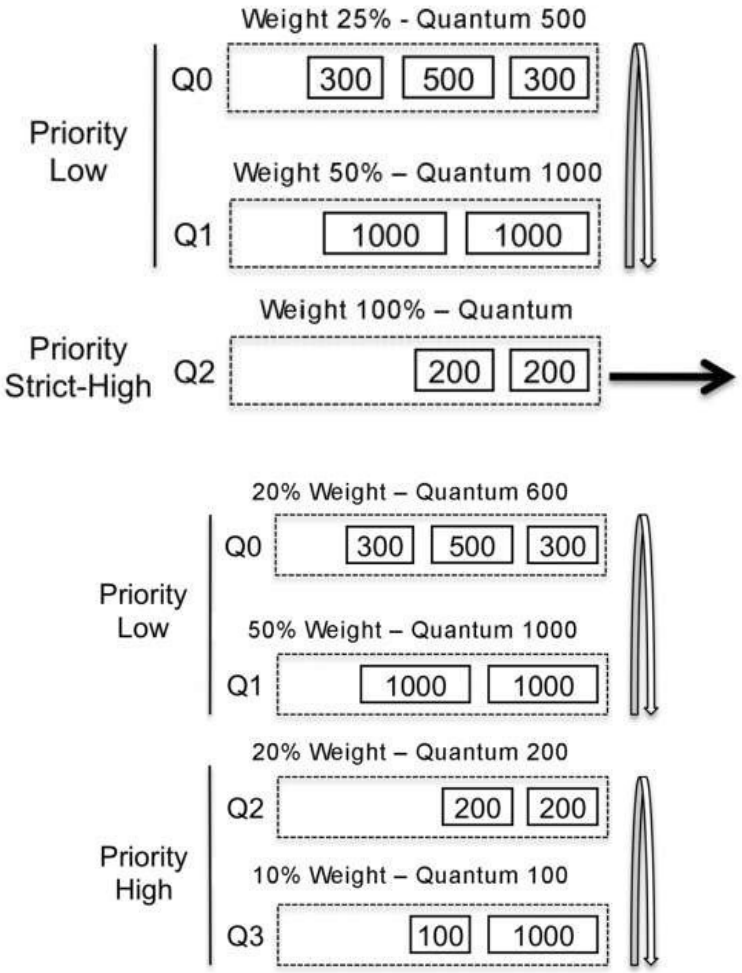
- Weighted round robin is similar to WFQ as it assigns different weights to different queues, but it considers packets rather than bits
  - Less precise in terms of respecting the weights, but OK in average
  - Less computationally intensive than WFQ



- Deficit weight round robin tries to bring back fairness regarding packet size to WRR without having to calculate the exact size:
  - It uses tokens (called quantums) which are assigned to queues proportionally to the weight at every turn
  - When a packet transmits it consumes quantums, in an amount that is proportional to the packet size in bytes
  - If not enough quantums are available the queue is skipped until the next round

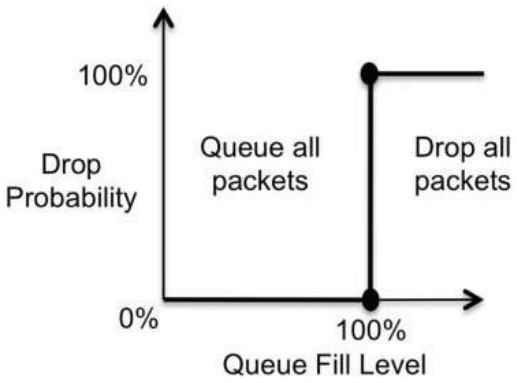
PB-DWRR

- Priority-based deficit weighted round robin (PB-DWRR) adds stricter priority for services that need very low delay and jitter
  - Some queues are served strictly before others while other queues follow DWRR
  - Issue if high-priority queue monopolise the capacity (same as PQ)
  - One solution is to police the high priority queue to some value so that packets to that queue are dropped if the exceed the limit

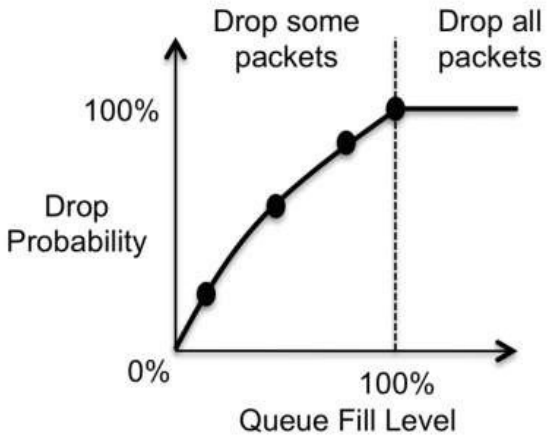


Other tools: RED

- We mentioned before that the dropper of a queue drops packets when the queue is 100% full (called tail drop)

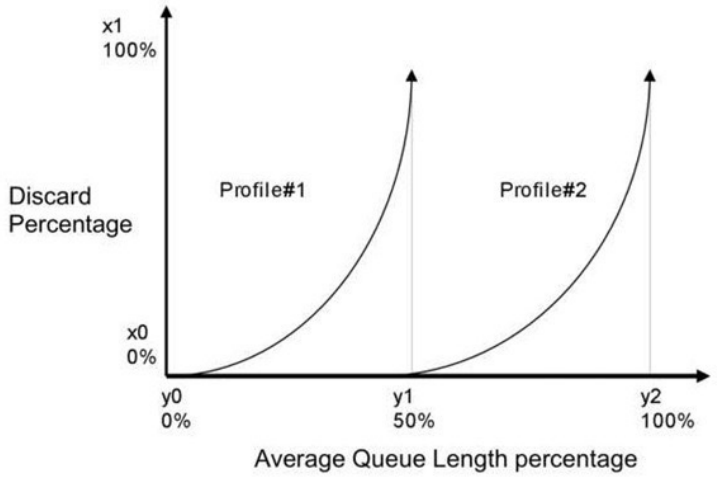


- However other strategies exist to drop packets before the queue gets full: Random Early Discard (RED)
  - RED drops packets with a certain probability as the queue starts filling up
  - The idea is to send TCP a “message” as TCP will react to a packet drop by lowering the transmission rate
  - Better to give an advanced notice than to drop suddenly a large amount of packets



Other tools: WRED

- Weighted Random Early Discard allow different drop profiles for different type of traffic:
  - It gives lower priority traffic
  a higher drop percentage



Example

Traffic	Class of service	Metering and policing			Queue	Scheduler	Egress rate	Rewrite
		Rate	Color	Action				
Black	COS1	<5M	Green	Accept	Use Q1	Prioritize Q1	Limit 8 Mbps store excess	None
		>5 M<7 M	Yellow	Accept				
		>7 M	Red	Drop				
White	COS2	<1 M	Green	Accept	Use Q2			Marking = X
		>1 M<2 M	Yellow	Accept				
		>2 M	Red	Drop				None